A Major Project report entitled

On

G.A.R.V.I.S (GPT-Assisted Robotic Voice Interactive System)

In partial fulfillment of the requirements for the award of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering (Data Science)

Submitted by

BANDRA ANUSHA (21E51A6704)

ADITHYA CHILUPURI (21E51A6707)

THAKUR ARYAN SINGH (21E51A6760)

V RAMESH (21E51A6762)

Under the Esteemed guidance of

MR. NAVA KISHORE

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Gowdavelly (Village), Medchal, Hyderabad, Telangana, 501401

(UGC Autonomous, Affiliated to JNTUH, Accredited by NAAC (A+) and NBA)

2024-2025

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(UGC Autonomous, Affiliated to JNTUH, Accredited by NAAC (A+) and NBA)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

CERTIFICATE

This is to certify that the Major Project entitled "**G.A.R.V.I.S (GPT-Assisted Robotic Voice Interactive System)**" is being submitted by **Bandra Anusha** bearing hall ticket number **21E51A6704**, **Adithya chilupuri** bearing hall ticket number **21E51A6707**, **Thakur Aryan Singh** bearing hall ticket number **21E51A6760**, **V Ramesh** bearing hall ticket number **21E51A6762,** in partial fulfillment of the requirements for the degree **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** by the Jawaharlal Nehru Technological University, Hyderabad, during the academic year 2024-2025. The matter contained in this document has not been submitted to any other University or institute for the award of any degree or diploma.

Under the Guidance of	**Head of the Department**
Mr.Nava Kishore	**Dr. M.V.A Naidu**
Associate Professor	**Professor & HoD**

Internal Examiner	**External Examiner**

HYDERABAD INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(UGC Autonomous, Affiliated to JNTUH, Accredited by NAAC (A+) and NBA)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

DECLARATION

We "**Bandra Anusha, Adithya chilupuri, Thakur Aryan Singh, V Ramesh**" students of '**Bachelor of Technology in CSE (Data Science)**', session: 2024- 2025, Hyderabad Institute of Technology and Management, Gowdavelly, Hyderabad, Telangana State, hereby declare that the work presented in this Major Project entitled **'G.A.R.V.I.S (GPT-Assisted Robotic Voice Interactive System)'** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

BANDRA ANUSHA (21E51A6704)

ADITHYA CHILUPURI (21E51A6707)

THAKUR ARYAN SINGH (21E51A6760)

V RAMESH (21E51A6762)

ACKNOWLEDGEMENT

An endeavor of a long period can be successful only with the advice of many well-wishers. We would like to thank our chairman, **SRI. ARUTLA PRASHANTH**, for providing all the facilities to carry out Project Work successfully. We would like to thank our Principal **DR. S. ARVIND**, who has inspired a lot through their speeches and providing this opportunity to carry out our Minor Project successfully. We are very thankful to our Head of the Department, **DR. M.V.A NAIDU** and B-Tech Project Coordinator **MS. RICHA TIWARI**. We would like to specially thank my internal supervisor **MR. NAVA KISHORE,** our technical guidance, constant encouragement and enormous support provided to us for carrying out our Major Project. We wish to convey our gratitude and express sincere thanks to all **D.C (DEPARTMENTAL COMMITTEE)** and **P.R.C (PROJECT REVIEW COMMITTEE)** members, non-teaching staff for their support and Cooperation rendered for successful submission of our Major Project work.

BANDRA ANUSHA (21E51A6704)

ADITHYA CHILUPURI (21E51A6707)

THAKUR ARYAN SINGH (21E51A6760)

V RAMESH (21E51A6762)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The goal of this project is to develop an intelligent and user-friendly virtual assistant named Garvis, designed to enhance everyday life. Inspired by existing virtual assistants like Cortana for Windows and Siri for iOS, Garvis aims to provide a seamless and natural interaction experience, whether through voice commands or keyboard input. Over time, Garvis learns from user interactions, tailoring its responses and actions to individual preferences and needs. Garvis excels at organizing schedules, setting reminders, and offering personalized recommendations and useful information. It assists with a wide range of tasks, including general conversations, web searches, retrieving images and videos, checking live weather conditions, providing word meanings, searching for medicine details, and offering health recommendations based on symptoms. By analyzing user commands with the help of machine learning, Garvis delivers optimal solutions efficiently. As a general-purpose desktop-based application, Garvis significantly enhances productivity by managing routine tasks and delivering timely information from online sources. Its smart and adaptive capabilities make it an indispensable tool for improving daily life and handling various activities effectively.

Keywords: Intelligent virtual assistant, Garvis, Voice commands, Personalized recommendations, Machine learning, Schedule organization, Web searches, Health recommendations, Productivity, Adaptive capabilities.

CHAPTER 1

1.1 INTRODUCTION

The Personalized Desktop Assistant is intelligent software designed to revolutionize the way individuals interact with their computers. It offers a tailored and user-centric experience that adapts to the unique preferences, habits, and workflows of each user. This personalized interface enhances productivity and user satisfaction, simplifying the complexities of desktop computing. The assistant organizes tasks and applications and learns and evolves alongside the user. As humans grapple with information overload, multitasking challenges, and the integration of various applications, the Personalized Desktop Assistant emerges as a beacon of efficiency. Its mission is to offer a central hub for desktop computing, taking care of the complex tasks and making it easier for users to focus on their work. In recent years, AI-powered assistants that can interact naturally with humans through voice, gestures, facial expressions, and other methods have become popular. The focus has shifted from humans self-learning to communicate with machines-to-machines self-learning to communicate with humans. These personalized assistants have been constantly improving and expanding beyond personal computers, establishing themselves on mobile devices and gadgets.

Figure 1.1.1: Voice Assistant

The first ever voice-activated product Radio Rex which was introduced in 1922, a simple toy dog that would jump out of its house when the user exclaimed its name. Today, automation is replacing human interaction rapidly, driven by advancements in technology such as machine learning and neural networks. Virtual assistants are software programs that can help with daily tasks, such as showing weather reports, giving news updates, and searching the internet, and they

can be controlled by voice commands. Voice-based intelligent assistants require an invoking word or wake word to activate the listener, followed by the command. Some well-known virtual assistants include Apple's Siri, Amazon's Alexa, and Microsoft's Cortana. This inspired the creation of a similar project, designed to be used efficiently on desktops Type Style and Fonts.

Computers have become very important devices and as well as less expensive over time. The idea behind the Personal Virtual Assistant is creation of an inexpensive, reliable and easy to use assistant. Virtual Assistant (VA) is a term that applies to computer-simulated environments that can simulate physical presence in places in the real world, as well as in imaginary worlds. Virtual Assistant is a real-time and interactive technology. It means that the computer can detect user inputs and modify the virtual world instantaneously. Interactivity and its captivating power contribute to the feeling of being part of the action on the environment that the user experiences. All human sensorial channels can be used to have a high-level interaction. Most current virtual assistant environments are primarily visual experiences, displayed either on a computer screen, but some simulations include additional sensory information, such as sound through speakers or headphones. Virtual Assistant technology has been a promising technology applicable in various domains of application such as training simulators, medical and health care, rehabilitation, education, engineering, scientific visualization, and the entertainment industry. The application works similarly to Siri/ Google Assistant. But the application deals with the computer itself mainly. A voice assistant is a digital assistant that uses voice recognition, natural language processing and speech.

In today's digitally-driven world, the majority of tasks are conducted electronically, with voice searches surpassing text searches and mobile web searches overtaking those done on computers. Analysts predict a further surge in voice searches, estimating that half of all searches will be voice-based by 2024. Virtual assistants have evolved significantly, now capable of handling tasks intelligently, including email management, intent detection, information extraction, process automation, and personalized responses. Researchers have increasingly focused on human activity recognition, with desktop virtual assistants programmed to assist with daily activities like checking the weather, setting reminders, and creating shopping lists. These assistants, accessible via text or voice commands, are primarily designed for desktop computer use, aimed at enhancing user productivity by automating routine tasks and sourcing information from online platforms. This project introduces a voice recognition system utilizing Natural Language Processing (NLP) algorithms to identify human activities. Voice communication enables users to interact seamlessly, with Automatic Speech Recognition (ASR) technology converting spoken words into computer-readable formats. Despite its advantages, distinguishing speech from background noise poses an additional challenge in audio processing.

The evolution of artificial intelligence has paved the way for innovative tools that enhance daily productivity and simplify complex tasks. Our project, G.A.R.V.I.S. (GPT-Assisted Robotic Voice Interactive System), represents a cutting-edge virtual assistant designed to revolutionize the way users interact with their desktop environments. Building on the foundational concepts of natural language processing (NLP), speech recognition, and machine learning, G.A.R.V.I.S. aims to provide a highly intelligent and user-friendly interface capable of managing diverse tasks effectively.

G.A.R.V.I.S. serves as a multi-functional assistant that not only executes commands but also adapts to individual user preferences over time, creating a personalized interaction experience. Whether through voice commands or keyboard inputs, the system is designed to seamlessly integrate into daily workflows, offering users a blend of convenience and efficiency. By focusing on adaptability and responsiveness, G.A.R.V.I.S. ensures a natural communication experience that aligns with the expectations of modern technology users.

Figure 1.1.2: Speech Recognition

With a robust foundation in Python programming and integration of advanced APIs, G.A.R.V.I.S. is equipped to handle a wide range of functionalities. These include organizing schedules, setting reminders, conducting web searches, retrieving multimedia content, checking live weather updates, and providing health recommendations. The system is further enhanced by its ability to analyze user inputs using GPT-driven machine learning algorithms, enabling it to deliver accurate and optimized responses in real time. The core objective of G.A.R.V.I.S. is to redefine the concept of a virtual assistant by offering an intuitive, versatile, and highly adaptive tool that supports users in managing routine and complex tasks alike. By leveraging state-of-the-art AI technologies, G.A.R.V.I.S. aims to become an indispensable part of daily life, facilitating greater productivity and an improved digital experience.

1.2. HISTORY OF VOICE ASSISTANT

In the 21st century, automation is rapidly changing human interaction for greater efficiency. The advancements in technology have led to innovations in Machine Learning, Neural Networks, and other such technologies that allow us to instruct machines to perform tasks or think like humans. One of the most significant innovations in this area is the development of digital assistants, such as Google Now, Siri, and Cortana, which allow users to interact with their computers by simply using their voice.

These digital assistants are particularly useful for elderly individuals, blind and physically challenged individuals, and children who might have difficulty interacting with their computers through traditional input methods. With the help of a voice assistant, even blind individuals can interact with their computers by using their voice alone.

The voice assistant described in this paper is a desktop-based assistant developed using Python modules and libraries. While it is capable of performing basic tasks such as checking weather updates, sending and checking emails, and opening applications, there is still a lot of room for improvement, particularly through the integration of Machine Learning and Internet of Things (IoT).

The future of digital assistants is such that they will be able to execute complex tasks and interact with the user at a more intuitive level, as seen in the fictional digital assistant, Jarvis, in the Iron Man movies. By using voice-activated digital assistants, there will no longer be a need to write long lines of code to perform tasks; the assistant will be able to do it for us. The desktop assistant can operate in supervised, unsupervised, or reinforcement learning modes, depending on its intended use. With the integration of IoT, the assistant will be able to interact with nearby smart devices and act as a single interface that controls everything in the surrounding environment. Through a successful digital assistant, it will be possible to control multiple devices around us with a single platform

From simple speech recognition software to advanced AIpowered virtual assistants, voice assistants have come a long way. The voice assistant journey, however; had started in the mid-20th century with crude attempts at speech recognition. By the 1950s Bell Labs had introduced “Audrey” then by the 1960s IBM came up with “Shoebox”. These served as a basis for further development. Nonetheless, these earliest systems were limited by technology and failed to gain widespread use due to their inadequacies and limited accuracy. This took place at the turn of the century when more advanced voice assistants were being developed. As early as 2001, Microsoft’s “Cortana” was launched, which led to Apple’s release of Siri on iPhone 4S in 2011. Introduced via iPhone 4S, Siri ensured that everyone bought into the virtual assistant concept through natural language command hence starting an era of voice-controlled interface. These are among some of the voice assistant platforms that have dominated the market such as Amazon’s Alexa in 2014 and Google Assistant released in 2016. These platforms used artificial intelligence (AI), natural language processing (NLP), and cloud computing so that they became increasingly personalized and responsive. Today there are voice assistants implemented across various devices or services thereby transforming how users interact with technology every da

Figure 1.2: History

Timeline of Mainstream Voice Assistants

The development and evolution of voice assistants have significantly shaped human-computer interaction in the modern era. The timeline presented in the figure showcases the launch dates of prominent voice assistant technologies introduced by major tech companies over the last decade:

- Apple Siri (October 2011): Siri was the first mainstream voice assistant, introduced by Apple for iOS devices. It pioneered voice-controlled personal assistance, enabling users to perform tasks like sending messages, setting reminders, and searching the web using natural language commands.

- Google Now (May 2012): Launched as part of the Android ecosystem, Google Now aimed to provide predictive and context-aware assistance based on user habits and search history. Although it was eventually phased out, it laid the foundation for future Google voice technologies.

- Microsoft Cortana (April 2014): Cortana was introduced as a virtual assistant for Windows-based systems. It was designed to help users manage tasks, set reminders, and interact with Microsoft services through voice.

- Amazon Alexa (November 2014): Amazon revolutionized the smart home industry with Alexa, integrated into the Echo line of devices. Alexa enabled users to control smart home appliances, play music, and access cloud-based services with voice commands.

- Google Assistant (November 2016): Building upon the capabilities of Google Now, Google Assistant marked a significant upgrade with conversational AI. It offers natural language processing, contextual understanding, and integration with a wide range of services and smart devices.

1.3 MOTIVATION

The motivation behind developing a desktop virtual assistant stemmed from a recognition of the evolving needs of users in navigating the digital landscape. While existing virtual assistants offer valuable assistance, there remains a notable gap in seamlessly integrating these functionalities into the desktop environment. Many solutions predominantly catered to mobile platforms, leaving desktop users with fragmented experiences and limited accessibility to intelligent assistance. Furthermore, the increasing complexity of tasks performed on desktop computers, coupled with the rising volume of digital information, underscored the necessity for a versatile and efficient virtual assistant tailored specifically for desktop use. My project seeks to address this gap by providing a comprehensive and intuitive desktop virtual assistant solution that seamlessly integrates into users' workflow. By developing a virtual assistant specifically optimized for desktop environments, I aim to enhance user productivity, streamline task management, and provide personalized assistance tailored to individual preferences and requirements. Moreover, I recognize the importance of natural language processing and voice recognition capabilities in facilitating seamless user interactions, thus incorporating advanced algorithms to ensure robust communication and comprehension.

Additionally, my motivation stems from a desire to democratize access to intelligent assistance, ensuring that desktop users, irrespective of their technical proficiency or device preferences, can benefit from the convenience and efficiency offered by virtual assistants. By bridging the gap between existing solutions and the unique demands of desktop computing, my project not only aims to empower users in optimizing their digital workflows but also to contribute to the advancement of human computer interaction research. Ultimately, my motivation lies in creating a more connected, efficient, and user-centric computing experience for desktop users worldwide.

Virtual assistants have become an integral part of our daily lives, providing us with instant access to information and enabling us to perform various tasks using natural language. However, the majority of existing virtual assistants are designed for mobile devices and lack the full functionality required for desktop computers. This gap in the market has led to an increasing demand for virtual assistants specifically designed for desktop computers. The motivation for this research project is to develop a virtual assistant for desktop computers that can perform a wide range of tasks using natural language processing and machine learning techniques. The proposed virtual assistant will be designed to assist users in performing tasks such as scheduling appointments, managing files, and retrieving information, among others. By providing users with a reliable and efficient virtual assistant, this project aims to improve productivity and enhance user experience on desktop computers. Additionally, the proposed virtual assistant will be designed to learn from user interactions and adapt to user preferences over time. This adaptive learning feature will enable the virtual assistant to provide personalized recommendations and perform tasks more efficiently, ultimately improving user satisfaction and increasing the user’s reliance on the virtual assistant.

CHAPTER 2

2.1 PROBLEM STATEMENT

Users face challenges in managing diverse tasks efficiently due to the limitations of existing virtual assistants, which often lack personalization, real-time optimization, and seamless integration of multiple functionalities.

There is also a need for enhanced interaction capabilities and better use of APIs for accessing and processing information. G.A.R.V.I.S. addresses these issues by integrating advanced AI technologies and APIs to deliver a personalized, adaptive, and efficient virtual assistant, capable of simplifying complex tasks and enhancing productivity

Design and develop a desktop assistant that can perform various tasks to assist users in their daily computer-related activities. The assistant should be able to understand voice and text commands, execute tasks efficiently, and provide relevant information to users. To develop a desktop voice assistant that enables users to interact with their computers through voice commands, reducing the need for manual typing, and enhancing overall desktop computing efficiency and user experience.

The current voice assistants use pattern recognition techniques, which are inaccurate, context-free, and prone to misunderstandings. It takes more time and money, and requires the internet. They also require database servers to store data, which increases the complexity of time and space. Additionally, they raise privacy hazards because when we offer commands, they are tied to the individual, such as his sleep patterns, banking information, address, and contacts, so the current system poses a threat to anyone's privacy. If we wish to add a custom command and result, for example, the command "tell my friend's name, results," Due to prebuilt commands, RAM is not supported

We all know about Cortana, Siri, Google Assistant and many other virtual assistants designed to help users of Windows, Android and iOS platforms succeed. But surprisingly, there isn't a complete virtual assistant for the Core Windows platform, which is made up of 70% of users. So unstable internet is a big problem for users who may have server issues and places where there is no internet access. The main purpose of creating self-help software (virtual assistant) is to use semantic information found on the web, users create content and provide information from information databases. The main purpose of Intelligent Virtual Assistant is to answer questions that the user may have. This can be done in a business environment, for example a business website with an interactive interface. Intelligent virtual assistants on mobile platforms, your voice to the user "What can I do for you?" It includes call-to-action programs that it asks. and then responds to feedback. A virtual assistant can be a huge time saver.

2.2 EXISTING SYSTEM

We know of many existing voice assistants that use word processing and speech recognition concepts such as Alexa, Siri, Google Assistant, Cortana. They listen to the commands given by the user according to their needs and perform that task effectively and efficiently. Because these voice assistants use intelligence, the results they provide are accurate and effective. These assistants can help reduce the effort and time people spend at work, eliminating the concept of typing and keeping track of other people we talk to and want a job with. These assistants are no less than assistants, but we can say that they are more efficient and effective in their work. Algorithms used to keep this group focused on the hard time and reduce the time. However, in order to use these assistants, it is necessary to have an account (e.g. Google account for Google Assistant, Microsoft account for Cortana) and only be used with an internet connection, because the assistants will use the internet connection. They include various devices such as phones, laptops, and speakers.

In the existing system, users need to be computer experts and provide precise instructions to find applications or files, which can be challenging. This wastes time as users must type specific details using a keyboard, increasing manual effort. There's a need for a more user-friendly and efficient solution for searching and interacting with the computer. Sure, here are the key points regarding existing systems for a desktop voice assistant with vision capabilities:

i. Lack of Pre-built Solutions: As of my last update in January 2022, there isn't a widely known pre-built system specifically designed for a desktop voice assistant with vision capabilities.

ii. Building from Existing Libraries: Developers typically build such systems using existing libraries and frameworks. OpenCV is commonly used for image processing and object detection, while SpeechRecognition handles speech-to-text conversion, and pyttsx3 is employed for text-to-speech synthesis.

iii. Integration and Customization: Developers integrate these libraries to create a cohesive system. Custom logic is implemented to process voice commands, analyze desktop images, and generate appropriate responses.

iv. Commercial Solutions: While general-purpose voice assistants like Amazon Alexa, Google Assistant, and Microsoft Cortana exist, they might not offer specialized support for desktop-specific vision tasks out-of-the-box.

v. Research and Academic Solutions: There might be specialized systems or frameworks developed within research institutions or academia to address specific requirements for desktop voice assistants with vision capabilities. These solutions are typically found through academic literature and research repositories.

2.3 PROPOSED SOLUTION

With the growing integration of artificial intelligence into everyday technology, virtual assistants have become indispensable tools for enhancing productivity and user experience. The Garvis Desktop Assistant project seeks to develop a personal AI assistant that can interact with users via voice commands to execute a range of tasks. Named after the iconic AI assistant from the Iron Man franchise, this assistant is designed to make desktop operations smoother and more intuitive by automating repetitive tasks and responding to user queries. The goal of this project is to create a system that leverages speech recognition, natural language processing (NLP), and task automation technologies. These capabilities enable the assistant to perform tasks such as managing files, sending emails, playing music, setting reminders, searching the web, and adjusting system settings—all through simple voice interactions. This hands-free functionality is intended to boost productivity and provide a more natural way of interacting with the computer. The system is primarily developed in Python, utilizing libraries like Speech Recognition for processing voice input, Pyttsx3 for text-to-speech output, and NLTK or Spacy for interpreting user commands. By integrating third-party APIs and custom modules, the assistant can provide responses tailored to the user's preferences and context. This introduction outlines the purpose and scope of the Jarvis Desktop Assistant, highlighting its potential to revolutionize the way users interact with their computers, similar to the rise of virtual assistants like Siri, Alexa, and Google Assistant.

The Desktop Assistant consists of, The proposed system Desktop Assistant project is like building the brain and communication skills for a friendly computer helper. It will understand what you say, talk back to you, and do tasks like fetching information or setting reminders. The system will have customizable features so it fits your preferences, and it will keep your information safe. The goal is to make your computer more accessible and helpful, but it's a bit like teaching a computer to understand and talk like a human, which can be tricky. Overall, this system could make working with your computer much easier and open up new possibilities for everyone.

Proposed Systems are:
1. Accessing user voice through microphone: Taking the input as speech patterns through microphone.
2. Voice recognition and conversion: Audio data recognition and conversion into text.
3. Matching voice with predefined keyword: Comparing the input with predefined commands.
4. Result: Giving the desired output.

The proposed concept presents an efficient implementation of a personal voice assistant. This system utilizes a Speech Recognition library that incorporates built-in functions to comprehend user commands and generate voice responses using Text-to-Speech functions. When the voice assistant captures the user's voice command, underlying algorithms are employed to convert it into text

2.4 SCOPE OF THE PROJECT

The project's scope includes developing a desktop assistant with features like opening Google, answering questions, searching and playing YouTube videos, sending messages, displaying maps, and providing date and time information. This involves voice recognition, NLP, voice activation, and seamless integration with desktop functions. Scope of a Desktop Assistant Project in brief points:

Voice Interaction: Responds to voice commands using speech recognition and text-to-speech.
Task Automation: Performs tasks like opening applications, checking emails, setting reminders, etc.
Web Search: Answers queries or fetches information using search engines or AI APIs.
Personalized Responses: Offers personalized suggestions based on user preferences or behavior.
System Monitoring: Provides system info like battery status, CPU usage, date/time, etc.
File Management: Can open, search, or organize files and folders on the system.
Integration Capabilities: Can integrate with APIs or tools like weather, news, calendar, or chatbot services.
GUI : May include a simple user interface for easy access and control.
Security and Privacy: Ensures user data is handled securely, with optional authentication features.
Extensibility: Designed in a modular way to add future functionalities easily.

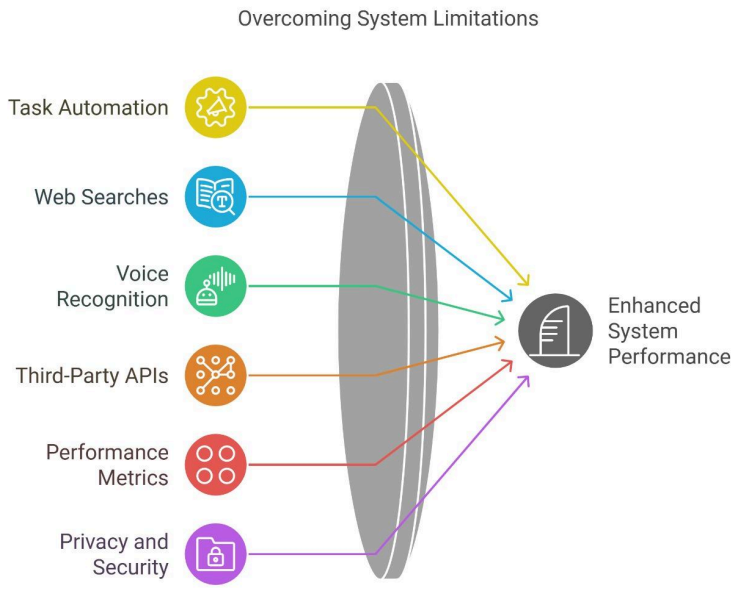


Figure 2.4: Scope of the project

2.5 OBJECTIVES OF PROJECT

The objectives of the G.A.R.V.I.S. project are as follows:

Enhanced Productivity: Provide users with a versatile virtual assistant capable of automating routine tasks and streamlining workflows to save time and effort.

Personalized User Experience: Leverage machine learning and GPT algorithms to adapt to individual user preferences and needs, ensuring a tailored interaction experience.

Comprehensive Functionality: Integrate features such as schedule organization, web searches, multimedia retrieval, and health recommendations to cater to diverse user requirements.

Seamless Interaction: Utilize NLP and advanced speech recognition to enable natural and intuitive communication through voice and keyboard inputs.

Real-Time Optimization: Deliver accurate and timely responses by analyzing user inputs and utilizing cutting-edge AI algorithms for efficient task execution.

CHAPTER 3

3.1 LITERATURE SURVEY

Voice assistants have become integral to modern technology, simplifying user interaction with devices and automating routine tasks. G.A.R.V.I.S. (GPT-Assisted Robotic Voice Interactive System) is inspired by advancements in AI, NLP, and speech recognition, aimed at enhancing productivity and offering a seamless interaction experience. This literature survey explores notable research and projects related to virtual assistants, highlighting their methodologies, functionalities, and applications.

"DESKTOP AI ASSISTANT: J.A.R.V.I.S Just a Rather Very Intelligent System (2014)"

This research presents a desktop AI assistant, JARVIS, capable of performing diverse tasks like setting reminders, conducting web searches, sending emails, and recognizing faces. The integration of Python, NLP, and Google's Speech Recognition API highlights the project's focus on task automation. Despite its strengths, the system's dependency on Python and third-party APIs limits its scalability and flexibility.

"JARVIS – The virtual Assistant(2022)"

The paper introduces JARVIS as a desktop-based application using voice recognition for task execution. It supports functionalities like news delivery, weather updates, and system operations. The waterfall development model ensures a structured approach, but the study lacks detailed performance metrics, comparative analyses, and discussions on user privacy.

"AI-Based Virtual Assistant Using Python: A Systematic Review(2023)"

This systematic review explores Python-based VPAs and their functionalities, including managing emails, conducting web searches, and providing reminders. It emphasizes Python libraries for NLP and speech recognition but notes the absence of performance metrics and privacy discussions, limiting its practical evaluation.

"Voice Assistant - A Review(2021)"

The paper focuses on a voice assistant designed for visually impaired users, leveraging text-to-speech and speech-to-text methods. It uses the Scrum methodology for development and emphasizes usability. While feedback from user testing was positive, the assistant's reliance on Python libraries and limited security measures present areas for improvement.

"Automating Desktop Tasks with a Voice-Controlled AI Assistant using Python(2024)"

This study develops an AI-powered virtual assistant to automate desktop tasks via voice commands. It integrates technologies like NLP and speech recognition, enabling functionalities such as Google searches, Gmail management, and interaction with WhatsApp and YouTube. The

project demonstrates robust task automation but could benefit from detailed performance evaluations and privacy considerations.

"Voice Assistant Using Python and AI"

This research paper explores various aspects of Python-based virtual voice assistants, delving into core concepts, operational methods, and applications of Python in developing virtual assistants.

"Voice-Based Virtual Personal Assistant Using Artificial Intelligence"

The paper emphasizes the importance of understanding existing literature on voice-based virtual personal assistants using AI in Python, highlighting the integration of speech synthesis, NLP, and AI for enhanced user interaction.

"JARVIS - Just A Rather Intelligent System"

The research model adopted in the design of JARVIS entails creating an AI virtual assistant that enhances activities using a mix of innovative technologies, focusing on ease of accomplishment.

Each of these studies underscores the transformative potential of virtual assistants while addressing limitations such as dependency on third-party APIs, lack of privacy features, and insufficient empirical evaluations. G.A.R.V.I.S. builds on these insights, aiming to overcome these challenges by integrating advanced APIs, enhancing user adaptability, and emphasizing privacy and security.

The literature survey reveals the ongoing advancements in the field of voice assistants, with a focus on virtual assistants like JARVIS and other Python-based systems. While these systems demonstrate impressive functionalities such as task automation, web searches, and voice recognition, common limitations include heavy reliance on third-party APIs, inadequate performance metrics, and concerns about user privacy and security. G.A.R.V.I.S. aims to address these shortcomings by leveraging advanced APIs, enhancing user adaptability, and ensuring robust privacy and security measures, ultimately contributing to the continued evolution of virtual assistant technology.

CHAPTER 4

4.1 REQUIREMENTS AND INSTALLATION

4.1.1 SOFTWARE REQUIREMENTS

The success and functionality of the Desktop Assistant depend largely on the software environment in which it is developed and executed. Below are the essential software requirements with detailed descriptions:

1. Operating System: Windows 10/11

Windows 10 is chosen as the preferred operating system for the development and deployment of the desktop assistant project. It offers a stable, user-friendly, and feature-rich environment that supports a wide range of development tools and libraries. With built-in features like Cortana, voice control APIs, and security modules, Windows 10 serves as an excellent platform for integrating voice-based functionalities and system-level automation. Its widespread usage and strong community support also make troubleshooting and compatibility testing more efficient.

2. Programming Language: Python (Latest Version)

Python is the core programming language used in the development of the desktop assistant due to its simplicity, readability, and vast ecosystem of libraries and frameworks. Python's versatility allows seamless integration of voice recognition, natural language processing, automation, and GUI elements. The latest version ensures access to improved features, enhanced performance, and the latest security updates. Python's cross-platform support and large community also aid in faster development and easier maintenance.

Key advantages of using Python include:

- Extensive support for third-party libraries.
- Built-in modules for system and file operations.
- Powerful tools for working with AI, voice recognition, and automation.
- Easy integration with APIs and databases.

3. Integrated Development Environment (IDE): Visual Studio Code (VS Code)

Visual Studio Code is the selected IDE for developing the desktop assistant project. It is a lightweight yet powerful source code editor that provides a robust environment for Python development. Features like intelligent code completion, syntax highlighting, debugging tools, and integrated terminal streamline the development process. Moreover, its vast extension marketplace allows easy installation of Python-specific tools, linters, formatters, and Git integration, thus enhancing productivity and code quality.

Benefits of using VS Code:

- User-friendly interface.
- Real-time error detection and debugging.
- Customizable themes and extensions.
- Excellent integration with version control tools like Git.

4. Tools: Python Libraries

A variety of Python libraries and modules are utilized to implement different functionalities of the desktop assistant, making the system intelligent, interactive, and responsive. Each library plays a vital role in handling specific tasks:

a.pyttsx3 & SpeechRecognition

These two libraries form the core of the assistant's voice interaction functionality:

- **pyttsx3** is a text-to-speech conversion library in Python. It allows the assistant to respond to user queries by speaking out the answers. Unlike other TTS libraries, it works offline and supports multiple voice engines.

- **SpeechRecognition** enables the assistant to listen to user commands by converting spoken input into text using various recognition engines like Google Speech API. It is lightweight and easy to implement, making it ideal for real-time voice-based applications.

Together, these libraries provide a smooth human-computer interaction experience through natural voice communication.

b.pyautogui

pyautogui is used for automating GUI interactions such as clicking, typing, scrolling, and taking screenshots. It enables the desktop assistant to simulate user actions, such as opening applications, typing messages, or controlling the mouse and keyboard. This is especially useful for tasks that require interaction with external software or desktop interfaces.

c.requests

The **requests** library is used for making HTTP requests to access data from the internet. Whether it's fetching the current weather, latest news, or triggering online services, requests provide a simple and effective way to handle API calls and web resources.

d.bs4 (BeautifulSoup)

BeautifulSoup (bs4) is a web scraping library used in conjunction with the requests module to extract data from HTML and XML documents. In the assistant, it is utilized to retrieve relevant information from websites, such as weather updates or latest headlines, and present them to the user in a readable format.

e.pywikihow

pywikihow allows the assistant to access step-by-step guides and instructions from WikiHow based on user queries. This helps the assistant answer "how-to" questions in an informative and structured manner. For example, queries like “How to make coffee?” can be answered with stepwise guidance fetched from WikiHow articles.

f.pywhatkit

pywhatkit is a versatile library used for performing a variety of tasks such as sending WhatsApp messages, playing YouTube videos, conducting Google searches, and more. This library adds rich functionality to the assistant by enabling it to interact with popular platforms and perform real-time online operations.

g.geopy & geocoder

These libraries are used for geolocation and mapping functionalities:

- **geopy** is used for geocoding addresses, calculating distances, and obtaining geographical data using various third-party APIs.
- **geocoder** is used to get the location of the user or IP address in real-time.

These tools help the assistant fetch location-based information such as the user's current city, coordinates, or nearest services.

```
pip install selenium
pip install pyttsx3
pip install SpeechRecognition
pip install pyautogui
pip install requests
pip install beautifulsoup4
pip install pywikihow
pip install pywhatkit
pip install geopy
pip install geocoder
pip install PyQt5
```

Figure 4.1: Software Installations

4.1.2 HARDWARE REQUIREMENTS

To ensure smooth functioning and reliable performance of the desktop assistant, certain hardware specifications are recommended. These components support the processing speed, memory, input/output devices, and audio-visual interaction necessary for a voice-enabled virtual assistant. Below is a detailed breakdown of the hardware requirements:

1.CPU(Central Processing Unit)

A basic small-core processor (CPU)would be enough for basic Ml tasks and smaller datasets.For larger and complex datasets, we can use a processor which is of high performance. We used Intel Core i5. The CPU is basically the brain of the computer and used for the execution of all tasks.

2.RAM(Random Access Memory)

For running basic Ml tasks and smaller datasets, we can use 8GB RAM which is more than enough. However, since our dataset is large and complex, we used 16GB RAM. RAM of large size ensures large data to be executed efficiently without any issues.

3. Storage and Capacity

SSD(Solid State Drive) - We used an SSD which plays a critical role in handling large datasets. They help in reading and writing the data at a faster pace.Based on the size of the dataset, the capacity keeps changing. We are using 256GB storage. Usually, it goes between 256 GB-512GB. This helps us have enough space for our models and datasets.

4. OS(Operating System)

An operating system is the most important for hardware management. It acts as an intermediary between the users and hardware , by providing an application for running the tasks. Ml frameworks can be run with Windows, Mac, and Linux operating systems; they are compatible with all three. We used a Windows operating system for our project.

5. Microphone

Microphones are essential hardware components for capturing sound and converting it into an electrical signal for processing. They are widely used in voice recognition systems, smart assistants, and communication devices. Microphones come in various types, including dynamic, condenser, and MEMS, each suited for different environments and applications. Their quality, sensitivity, and directional properties significantly impact the performance of audio-based systems.

TECHNOLOGY USED IN PERSONAL DESKTOP VOICE ASSISTANT

The desktop voice assistant is a software system that enables users to perform various tasks using voice commands. The system should be able to recognize voice commands accurately, respond promptly, and perform the requested tasks efficiently.

Functional Requirements:

The desktop voice assistant should have the following functionalities:

• **Wake word detection**: The system should be able to detect a wake word such as "Hey, assistant" to activate the assistant.

• **Voice recognition**: The system should be able to accurately recognize and interpret voice commands from the user.

• **Natural language processing:** The system should be able to understand the user's intent and respond accordingly.

• **Task execution**: The system should be able to execute tasks such as setting reminders, playing music, sending emails, and searching the web.

Non-functional Requirements

The desktop voice assistant should meet the following quality attributes:

• **Accuracy**: The system should have a high accuracy rate in recognizing voice commands. • Response time: The system should respond promptly to user commands. • Security: The system should be secure and protect user privacy.

• **Usability**: The system should be easy to use and have a user-friendly interface.

• **Constraints**. The desktop voice assistant should work on Windows, Mac, and Linux operating systems. The system should also be compatible with various microphones and audio input devices.

• **Assumptions and Dependencies:** The desktop voice assistant assumes that the user has a reliable internet connection for certain tasks such as web searches and email sending. The system also depends on third-party APIs for certain functionalities such as weather updates and music streaming.

CHAPTER 5

5.1 BLOCK DIAGRAM

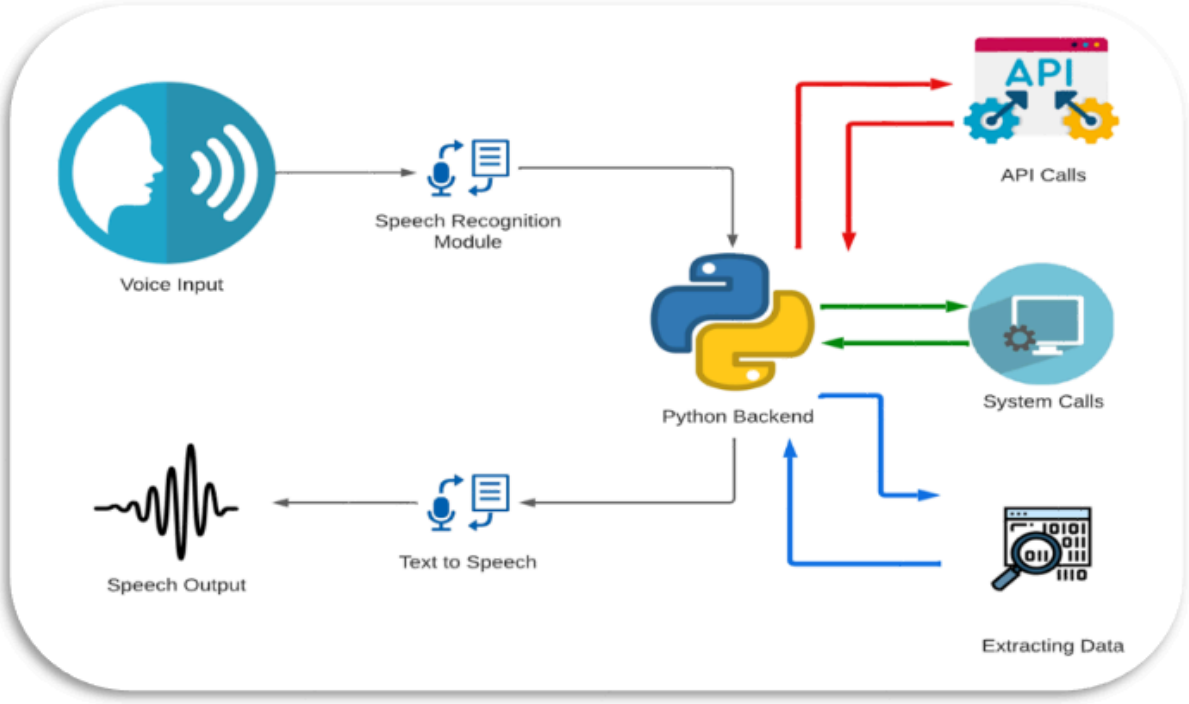


Figure 5.1: Block Diagram

The detailed block diagram illustrated in Figure presents the workflow of the desktop voice assistant system, outlining the complete process from receiving user voice input to delivering the final speech output. This modular architecture ensures seamless integration of various components that work together to carry out intelligent and automated tasks based on voice commands.

1. Voice Input

The workflow begins with the user providing a voice command through a microphone. This audio input acts as the primary interface for interaction with the desktop assistant.

2. Speech Recognition Module

Once the voice input is captured, it is passed to the Speech Recognition Module. This module converts the spoken language into textual data using libraries such as Google Speech Recognition, Vosk, or CMU Sphinx. The transcribed text is then sent to the Python backend for further processing.

3. Python Backend (Core Logic)

The Python backend serves as the central processing unit of the system. Based on the recognized text, the backend interprets the intent of the command and initiates relevant actions. These actions include:

- API Calls (Red Arrow): For queries like weather information, news updates, or searching the web, the backend initiates API calls to fetch data from third-party sources like OpenWeatherMap, NewsAPI, or custom GPT integrations.

- System Calls (Green Arrow): Commands like opening applications (e.g., calculator, notepad, browser), controlling system settings, or performing file operations are executed using system-level commands (e.g., os.system, subprocess).

- Data Extraction (Blue Arrow): For commands that require reading local files, fetching data from databases, or performing text summarization, the backend accesses and processes the required data.

4. Text to Speech (TTS)

After the action is completed or a response is generated, the Text to Speech (TTS) module converts the textual response into audible speech using engines like pyttsx3 or gTTS. This makes the interaction more human-like and accessible.

5. Speech Output

Finally, the response is delivered back to the user in the form of speech output, completing the interaction cycle.

This modular design ensures that the system is scalable, maintainable, and highly interactive, leveraging both machine intelligence and natural language processing capabilities to deliver a smart desktop assistant experience.

CHAPTER 6

6.1 METHODOLOGY

The methodology adopted in developing the Desktop Assistant involves a systematic approach combining software design principles, speech processing, GUI development, automation, and web integration. This section explains how different components were integrated to deliver a fully functional and interactive assistant capable of understanding and executing user commands. This project aims to develop a voice-controlled assistant using Python, which involves several phases, each focusing on specific tasks to ensure smooth interaction and execution of commands. The core components of the system are:

6.1.1 REQUIREMENT GATHERING AND PLANNING

The first step was to understand the core functionalities required in a virtual desktop assistant. Based on real-world assistant systems like Cortana, Siri, and Google Assistant, a feature list was prepared. The assistant was expected to:

- Understand voice commands
- Respond via speech
- Automate desktop tasks
- Fetch real-time data from the web
- Provide a user-friendly GUI

The planning stage also involved selecting appropriate tools and libraries such as Python, PyQt5, Selenium, and speech recognition modules to fulfill the desired functionalities.

6.1.2 DESIGN AND ARCHITECTURE

The assistant was designed as a **monolithic desktop application** with modular Python code to handle various tasks. The overall architecture includes:

- **Voice Interface Module**: For speech input and output
- **Command Processing Engine**: For interpreting commands and executing logic
- **Automation Layer**: To control applications and execute OS-level tasks
- **Web Interaction Module**: For fetching external data
- **Graphical User Interface (GUI)**: Built with PyQt5 for interactive user controls

The design emphasized both command-line and graphical interactions to make it accessible for all users.

6.1.3 VOICE RECOGNITION AND TEXT TO SPEECH INTEGRATION

Using the speech recognition library, the system was trained to capture and interpret voice inputs from the microphone. Once converted into text, the command is processed using logic built into the assistant.

Simultaneously, the assistant responds to the user using the pyttsx3 library, which converts text responses into speech. This two-way interaction allows users to control the system hands-free.

6.1.4 TASK AUTOMATION

The pyautogui library was used for desktop automation. This includes actions such as:

- Opening applications like Chrome, Notepad, or system utilities
- Taking screenshots
- Typing or clicking automatically

Custom command rules were written to associate voice instructions with specific automation tasks.

6.1.5 WEB INTEGRATION AND DATA FETCHING

To provide real-time services like weather updates, news headlines, or search results, two main approaches were used:

1. **API Integration**:
    - APIs from services like OpenWeatherMap or Wikipedia were used to fetch structured data.
    - The requests library was used to call and handle these APIs.
2. **Web Scraping**:
    - For websites without APIs, beautifulsoup4 and selenium were used to extract content dynamically.
    - Selenium enabled browser automation for services like YouTube or Google search.

6.1.6 GUI DEVELOPMENT

A user-friendly graphical interface was designed using **PyQt5** and **Qt Designer**. This included buttons, labels, and command displays to visually represent tasks performed by the assistant.

The GUI was integrated with the backend logic using Python signals and slots, enabling real-time updates and feedback within the application

6.1.7 TESTING AND OPTIMIZATION

Each module was tested independently and in integration. The following aspects were validated:

- Voice accuracy and noise handling
- Response correctness
- GUI responsiveness
- Automation timing
- Error handling for unavailable services or incorrect commands

Optimization involved tuning the command recognition sensitivity and refining the assistant's response logic to handle a wider range of natural language inputs.

6.1.8 FINAL INTEGRATION

All modules - voice, automation, web, and GUI - were brought together in the main Python script. This allowed seamless user interaction from the start of the assistant to the execution of user requests. The application was tested on a Windows 10 environment with Python 3.x and VS Code as the development IDE.

The methodology ensured that the desktop assistant was built in a structured, efficient, and scalable manner. The use of open-source libraries and modular programming helped in rapid development, while real-time feedback through both GUI and voice made the assistant user-friendly and interactive.

Through these phases, the system efficiently processes voice commands, converts them into actionable tasks, and responds in a manner that enhances user interaction, all while functioning offline.

Figure 6.1 : Methodology

CHAPTER 7

7.1 WORKING

The working of the Desktop Assistant is based on real-time voice command processing, speech synthesis, web automation, and user interface synchronization. The assistant performs tasks in a seamless pipeline that converts user speech into actions and responds with visual and auditory feedback.

Step-by-Step Execution Flow

1. Voice Command Input

- The interaction begins with the user providing a **voice command** through the system's microphone.
- The assistant constantly listens for input using the **speech_recognition** library.
- Once the voice is detected, it captures the audio in real-time and sends it for further processing

2. Speech-to-Text Conversion

- The captured audio is converted into a string format using the **Google Speech Recognition API**, which is accessed via the speech_recognition library.
- This transformation is crucial as it translates human speech into machine-readable text (query).
- For instance, if the user says *"Open YouTube"*, the assistant recognizes and converts it to the string "open youtube".

3. Command Parsing and Decision Making

- The string query is passed to the **core command processing engine**, which matches the command to predefined operations.
- A series of **conditional statements** (if-elif blocks or dictionaries) evaluate the command.
- Depending on the type of command, the assistant takes appropriate action. For example:
    - **Web automation tasks** are handled using **Selenium WebDriver** (e.g., opening websites, logging in).
    - **Web scraping tasks** like fetching news or weather are handled by **BeautifulSoup**.
    - **System-level operations** such as opening files, taking screenshots, or playing music use the os and pyautogui libraries.

4. Task Execution

- Once the command is interpreted, the assistant carries out the task using the appropriate module.

- Examples include:

    - Playing a song on YouTube via pywhatkit
    - Searching Wikipedia using pywikihow
    - Automating browser using selenium
    - Fetching geolocation using geopy and geocoder

5. Generating and Delivering Output

- After the task is completed, a **response string** is generated to confirm the action or provide information.
- This string is then passed to the **speak() function**, which uses the pyttsx3 library to **convert the text to speech** and relay it back to the user through audio.

6. GUI Update and Parallel Display

- Simultaneously, the output message and task status are **displayed on the graphical interface** developed using PyQt5.
- The GUI updates occur in real-time, ensuring the user can **visually track** what the assistant is doing.
- This dual feedback mechanism (audio + visual) makes the assistant more interactive and user-friendly.
- The GUI also refreshes every second using timers to provide a smooth experience with dynamic updates like:

    - Real-time clock
    - Running logs of user commands
    - System status

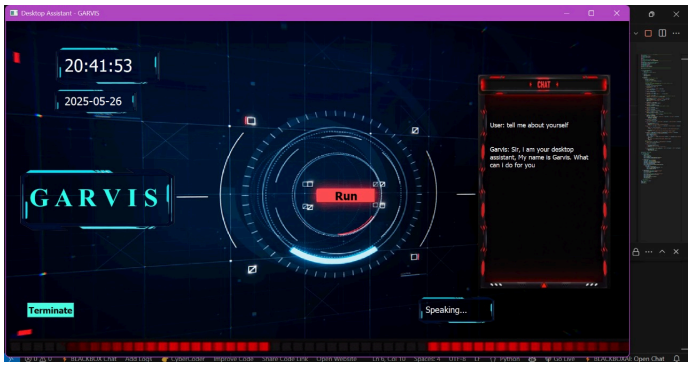


Figure 7.1: GUI

7.2 CODE

```
#--------------------------------For Features-------------------------------------
import pyautogui
from features.basic import *
from features import walter
from features.sense import *
import os
import sys
from features.search_web import findAns
from features.win_automate import WindowAutomate
#--------------------------------For GUI ------------------------------------------
import PyQt5
from PyQt5.QtCore import QTime, QTimer, QDate, Qt
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.uic import loadUiType
from Walter_UI import Ui_Garvis
#-------------------------------------------------------------------------------
import pyjokes #for jokes
#-------------------------------------------------------------------------------

class MainThread(QThread):
    def __init__(self) -> None:
        self.x = 1
        super(MainThread, self).__init__()

    def run(self):
        self.obj = walter()
        self.obj.wishuser()
        self.task()

    def task(self):
        # running the while loop infinite times
        while True:
            self.query = takecomand()
            # if user chats (conversation)
            chatresponse = self.obj.chatwalter(self.query)

            # if not in chatbot
            # greet and perform task simultaneously - (can run multiple command at once)
            # eg- hello walter, what is the temperature?
            if chatresponse == 0:
```

```
    self.query = self.obj.efficient(self.query)

if (WindowAutomate(self.query)!= 0):
    print("Done !")

elif 'open' in self.query or 'launch' in self.query:
    self.obj.open(self.query)

elif 'close' in self.query or 'terminate' in self.query:
    self.obj.close(self.query)

elif 'the date' in self.query:
    speak("Today is " + self.obj.day() + ', ' + self.obj.date())

elif 'the time' in self.query:
    # declaring the strTime variable to  get the current time according to mearidain
    speak("Sir, The current time is " + self.obj.time())

elif 'screenshot' in self.query or 'take a screenshot' in self.query:
    cwd = os.getcwd()
    pyautogui.screenshot(cwd + r'\image\Image' + str(self.x)+'.png')
    speak("Screenshot is saved as Image" + str(self.x))
    self.x += 1

elif 'temperature' in self.query:
    speak(self.obj.temperature(self.query))

elif "weather" in self.query:
    speak(self.obj.weather(self.query))
    # chat.append("Walter: " + chatmsg2) #prints weather in chatbox

elif "how to" in self.query:
    speak(self.obj.howto(self.query))

elif "search" in self.query:
    speak(self.obj.web_search(self.query))

elif "near" in self.query or 'nearby' in self.query:
    speak(self.obj.near(self.query))
    # chat.append("Walter: "+ nearby(self.query))   #adding msg to chatbox

elif "joke" in self.query or 'jokes' in self.query:
    speak(pyjokes.get_joke())
    self.query = takecomand()
    while 'one more' in self.query or 'another one' in self.query or 'once more' in self.query:
        speak(pyjokes.get_joke())
```

```
            self.query = takecomand()

        elif 'send mail' in self.query:
            self.obj.send_mail(self.query)

            elif 'join meet' in self.query or 'create a meet' in self.query or 'create a new meet' in self.query or 'join my class' in self.query:
            try:
                self.obj.join_meet(self.query)
            except Exception as e:
                speak("Sorry sir. I am not able to join meet right now")

            elif "my location" in self.query or "where am i" in self.query or "current location" in self.query:
            try:
                ci, st, co = self.obj.my_location()
                speak(
                    f"Sir, your current location is {ci} city which is in {st} state and country {co}")

            except Exception as e:
                speak("Sorry sir, I coundn't fetch your current location. Please try again")

        elif "where is" in self.query or 'location of' in self.query or 'distance of' in self.query:
            target_loc, distance, place = self.obj.location(self.query)
            # city = target_loc["city"]
            state = target_loc["state"]
            country = target_loc["country"]
            try:
                    res = f"{place} is in {state} state of country {country}. It is {distance} km away from your current location"
                speak(res)
            except:
                res = "Sorry sir, I couldn't get the location. Please try again"
                speak(res)

        #battery status
        elif "battery status" in self.query or "remaining battery" in self.query:
            speak(str(self.obj.battery_status()))

        elif "set alarm" in self.query:
            try:
                self.obj.set_alarm(self.query)
            except Exception as e:
                speak("Sorry sir, i am not able to set the alarm.")

            elif 'who' in self.query or 'what' in self.query or 'when' in self.query or 'where' in
```

```
self.query or 'how' in self.query or 'why' in self.query or 'which' in self.query:
            #finding answers from API/web/wikipedia
            if chatresponse == 0 :
               speak(findAns(self.query))


startexecution = MainThread()

class Main(QMainWindow):
   def __init__(self):
      super().__init__()
      self.ui = Ui_Garvis()
      self.ui.setupUi(self)
      self.ui.run_button.clicked.connect(self.starttask)
      self.ui.Terminate.clicked.connect(self.close)

   def showText(self):
      current_time = QTime.currentTime()
      current_date = QDate.currentDate()
      lable_time = current_time.toString('hh:mm:ss')
      lable_date = current_date.toString(Qt.ISODate)
      self.ui.Date.setText(lable_date)
      self.ui.Time.setText(lable_time)
      self.ui.state_of_assistant.setText(listToString(state))
      for item in chat:
         global chat_prev
         if len(chat) != len(chat_prev):
            self.res = listToString(chat[len(chat) - 1])
            self.ui.Chat_box.append(self.res)
            chat_prev.append(self.res)

   def starttask(self):
      self.ui.movie = QtGui.QMovie("image/Walter bg.gif")
      self.ui.bg_lab.setMovie(self.ui.movie)
      self.ui.movie.start()

      self.ui.movie = QtGui.QMovie("image/footer(line).gif")
      self.ui.footer_img.setMovie(self.ui.movie)
      self.ui.movie.start()
      timer = QTimer(self)
      timer.timeout.connect(self.showText)
      timer.start(1000)
      startexecution.start()

   # def showquery()
```

```
app = QApplication(sys.argv)
friday = Main()
friday.show()
exit(app.exec_())
```

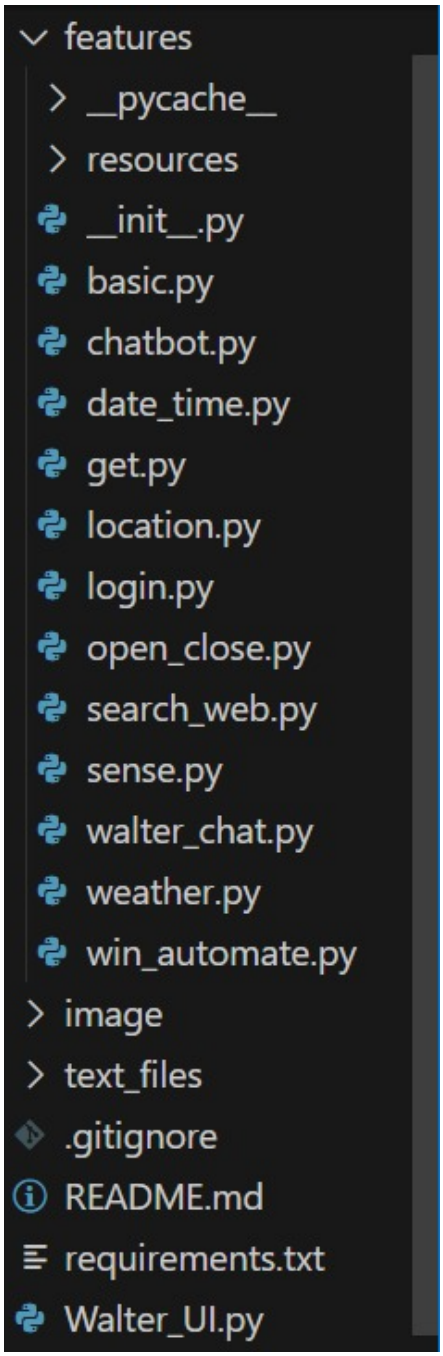


Figure 7.2: Features

7.3 FEATURES

The Desktop Assistant is designed to handle a wide range of tasks that support daily computer usage and enhance productivity through voice-based automation. Below is a detailed overview of the key features implemented in the system:

1. Sending Emails

- Users can send emails to any recipient by simply speaking the message and recipient's address.
- The assistant uses **SMTP protocol** to connect with email servers and deliver messages securely and reliably.

2. Voice-Controlled Web Browsing

- The assistant can **crawl or search any website** based on voice commands.
- For example, a command like *"Search Python tutorials"* will open relevant search results using **Selenium WebDriver** or pywhatkit.

3. Real-time Weather Updates

- The assistant fetches live weather information using online APIs or web scraping.
- It provides details such as temperature, humidity, weather conditions, and location-based forecasts through both voice and GUI.

4. Launching Desktop Software

- Users can open desktop applications such as **Notepad, Chrome, File Explorer**, or custom programs using voice commands.
- This is achieved via the os and subprocess libraries for local system control.

5. Note Creation and Saving

- The assistant can **create and save notes** as text files.
- Users can dictate content, and the system will save it automatically with a timestamped filename.

6. System Statistics Monitoring

- The assistant provides system-level information like:
    - CPU usage
    - RAM consumption
    - Battery percentage
- This is done using libraries like psutil, enhancing system awareness and resource tracking.

7. Voice-Based Query Searching

- It allows users to **search any general or technical query** by converting speech to text and fetching relevant results from Google or Wikipedia.
- For example, *"What is Machine Learning?"* triggers a search and returns summarized answers.

8. Location & Maps Integration

- Using libraries like geopy and geocoder, the assistant can:
    - Detect current location
    - Open maps for navigation
    - Share location-based info like nearby places

9. Daily Utility Functions

- A collection of small but useful day-to-day tasks are supported, including:
    - Telling jokes
    - Playing music on YouTube
    - Taking screenshots
    - Setting reminders or alarms
    - Locking or shutting down the system
    - Sending WhatsApp messages via pywhatkit

10. Real-time Interactive GUI

- An engaging PyQt5-based GUI updates live to show the assistant's actions, status, system info, and real-time clock.
- This enhances the user experience and provides visual confirmation of tasks being executed.

7.4 ADVANTAGES

Some advantages of Personal Desktop Voice Assistant include:

• Make up private conversations: With voice assistants, marketers have a new opportunity to start conversations in a more personalised way. With voice assistants, users typically express exactly what they want and are thinking. As a result, the channel enables marketers to respond with what they require and then keep in touch with customers to provide a customized experience.

• Reach a large number of users at once: Marketers can reach several consumers in a single home thanks to voice assistants. Because each of these customers has different brand preferences, product interests, and music playlists, they all make distinctive purchasing judgements. One voice assistant allows marketers to collect more data and sell through a single targeted campaign, resulting in better outcomes.

• Beyond the typical tools The fact that voice assistants are gaining popularity outside of our homes and mobile devices is an additional benefit. They are becoming more prevalent in our automobiles, smart Televisions, wearable technology, and home appliances. They offer fresh chances to accomplish even more goals while also giving current clients more value.

• Achieve elusive prospects You have a better chance of reaching your target demographic by marketing via a virtual assistant on smartphones or smart speakers. It provides you with an alternative to doing so to the Internet and mobile. Voicebot.ai estimates that 87.7 million American people currently use smart speakers, an increase of 32 percent from January 2019 and an increase of 85 percent.

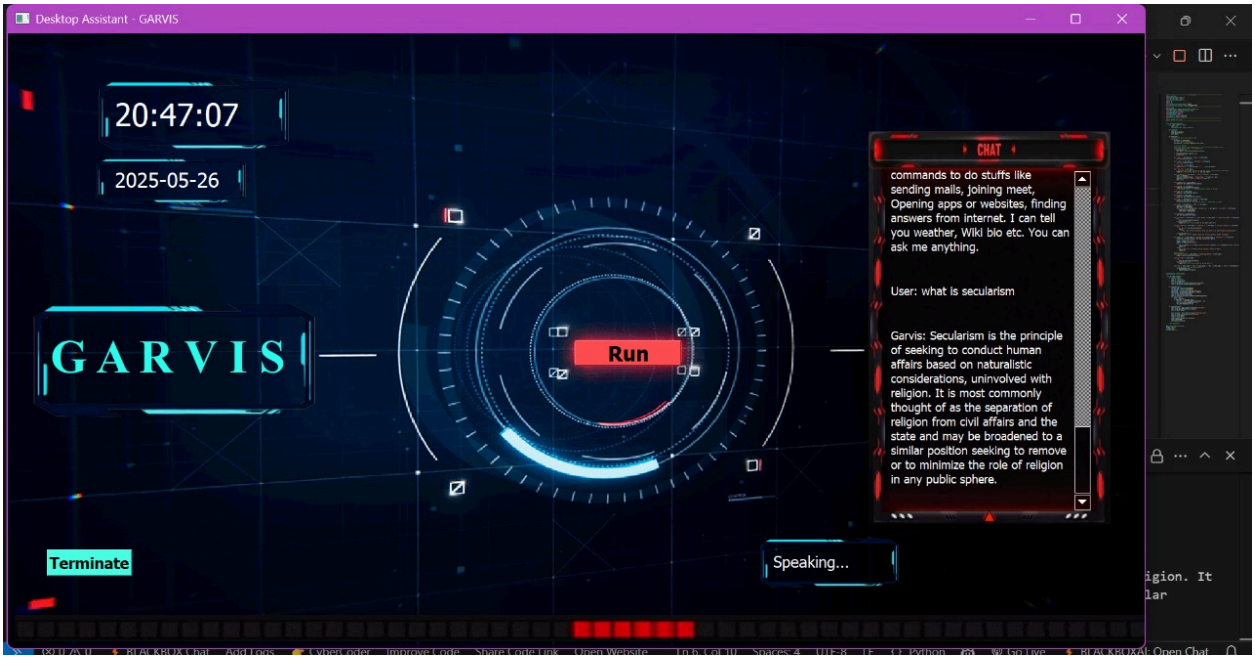
These are only a few instances of the jobs that voice assistants can complete; we can conduct a great deal more depending on our needs. Voice assistants' capabilities and advancements are always growing day by day to give users improved performance. Our desktop-based voice assistant is built using Python modules and libraries, allowing it to function quickly and efficiently on the desktop.

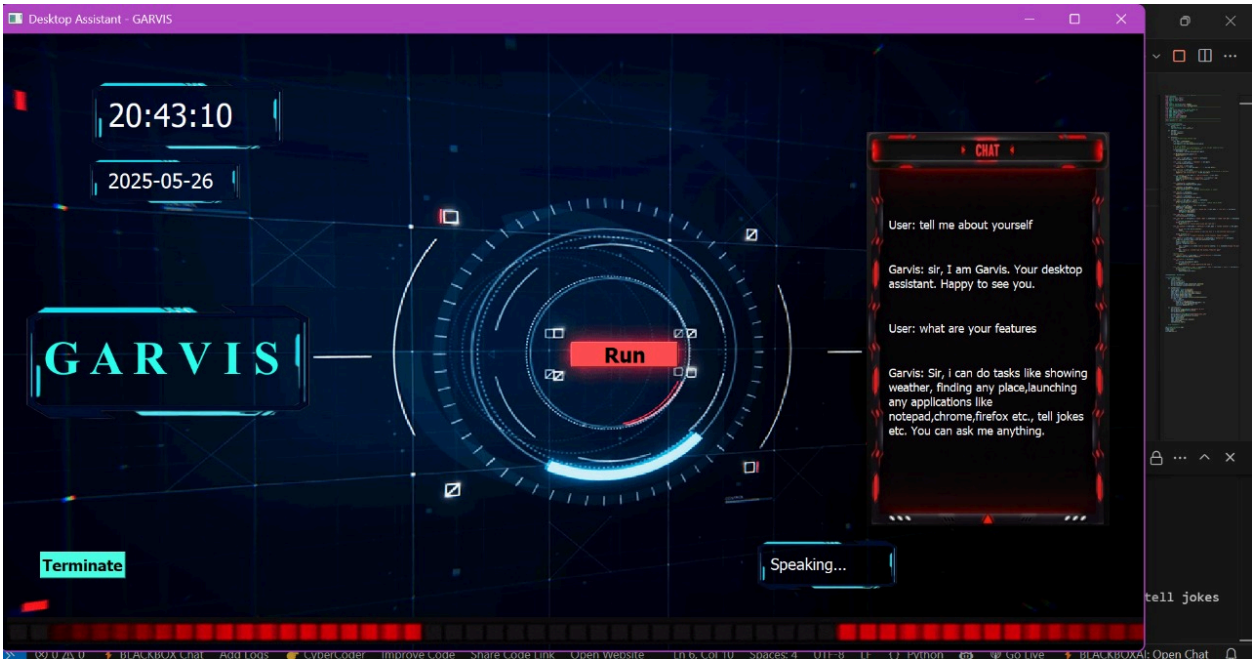
To make it possible for a very engaging user experience: Like no other interface, voice help keeps users' attention. Users can ask for anything they want by speaking naturally to the programmes.

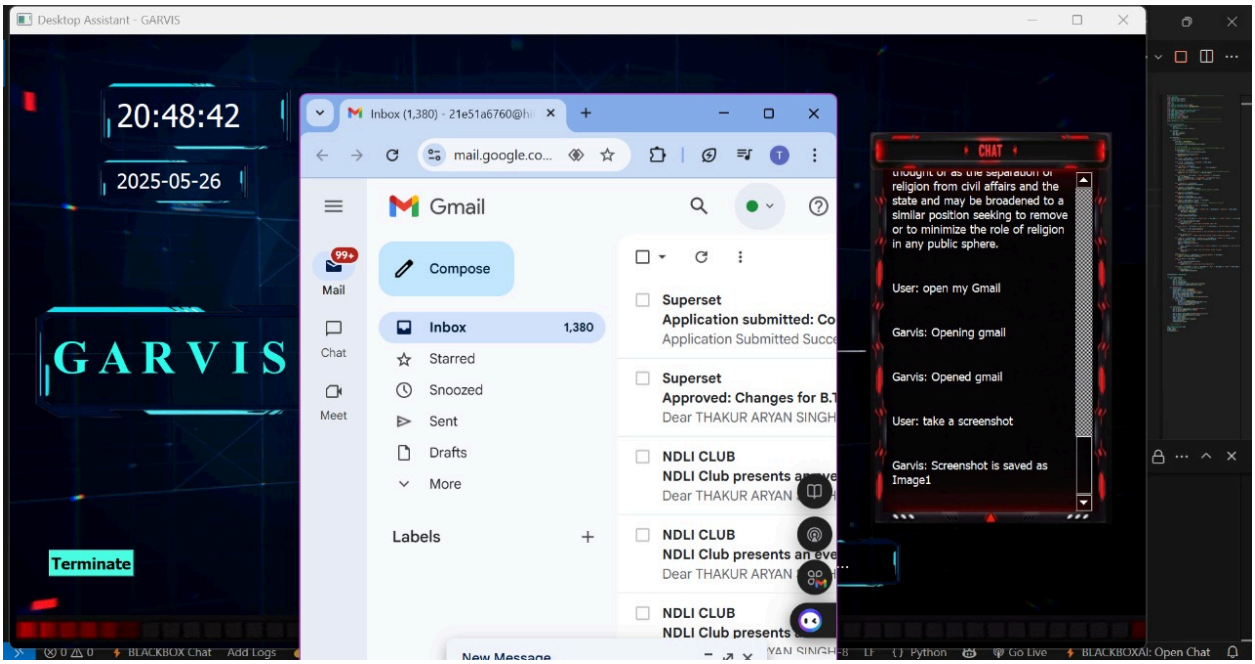
To eliminate user annoyance with the application: With the current machine system, we must touch, type, and use a mouse to complete our task, which occasionally causes user frustration. Users can ask their desired task directly utilising a voice assistant.

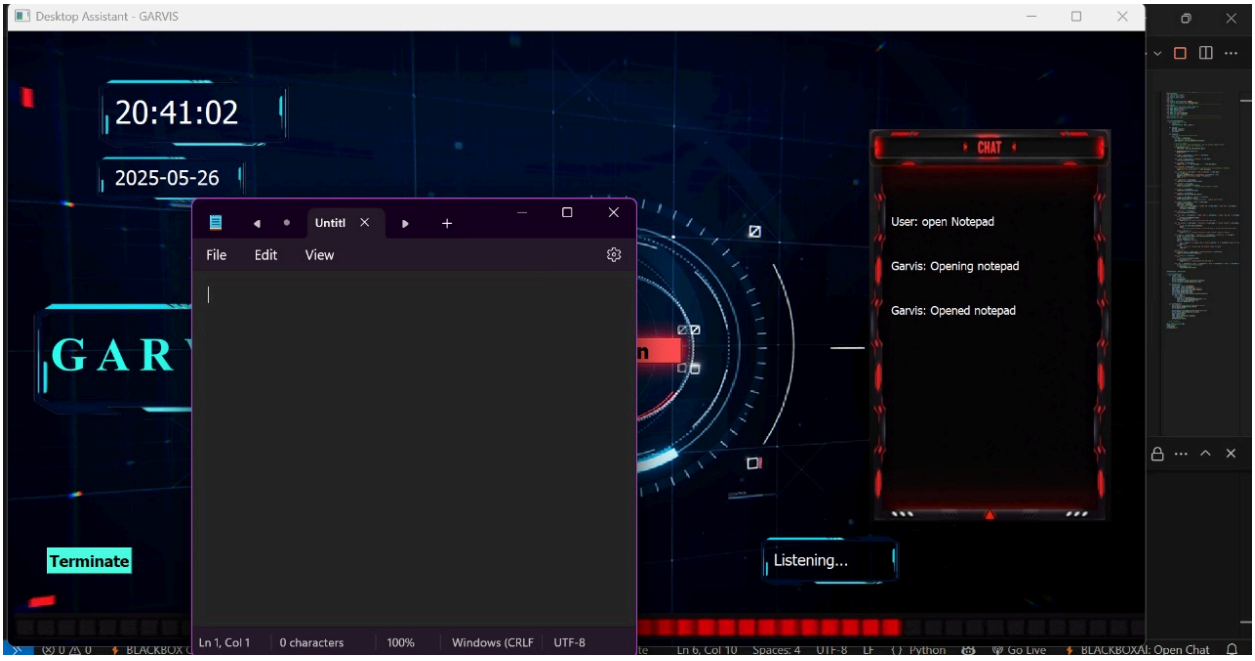
CHAPTER 8

8.1 RESULTS






Desktop Assistant - GARVIS
20:48:42
2025-05-26
GARVIS
Terminate
Gmail
Compose
Inbox 1,380
Starred
Snoozed
Sent
Drafts
More
Labels
CHAT
User: open my Gmail
Garvis: Opening gmail
Garvis: Opened gmail
User: take a screenshot
Garvis: Screenshot is saved as Image1


Desktop Assistant - GARVIS
20:41:02
2025-05-26
Terminate
File
Edit
View
CHAT
User: open Notepad
Garvis: Opening notepad
Garvis: Opened notepad
Listening...

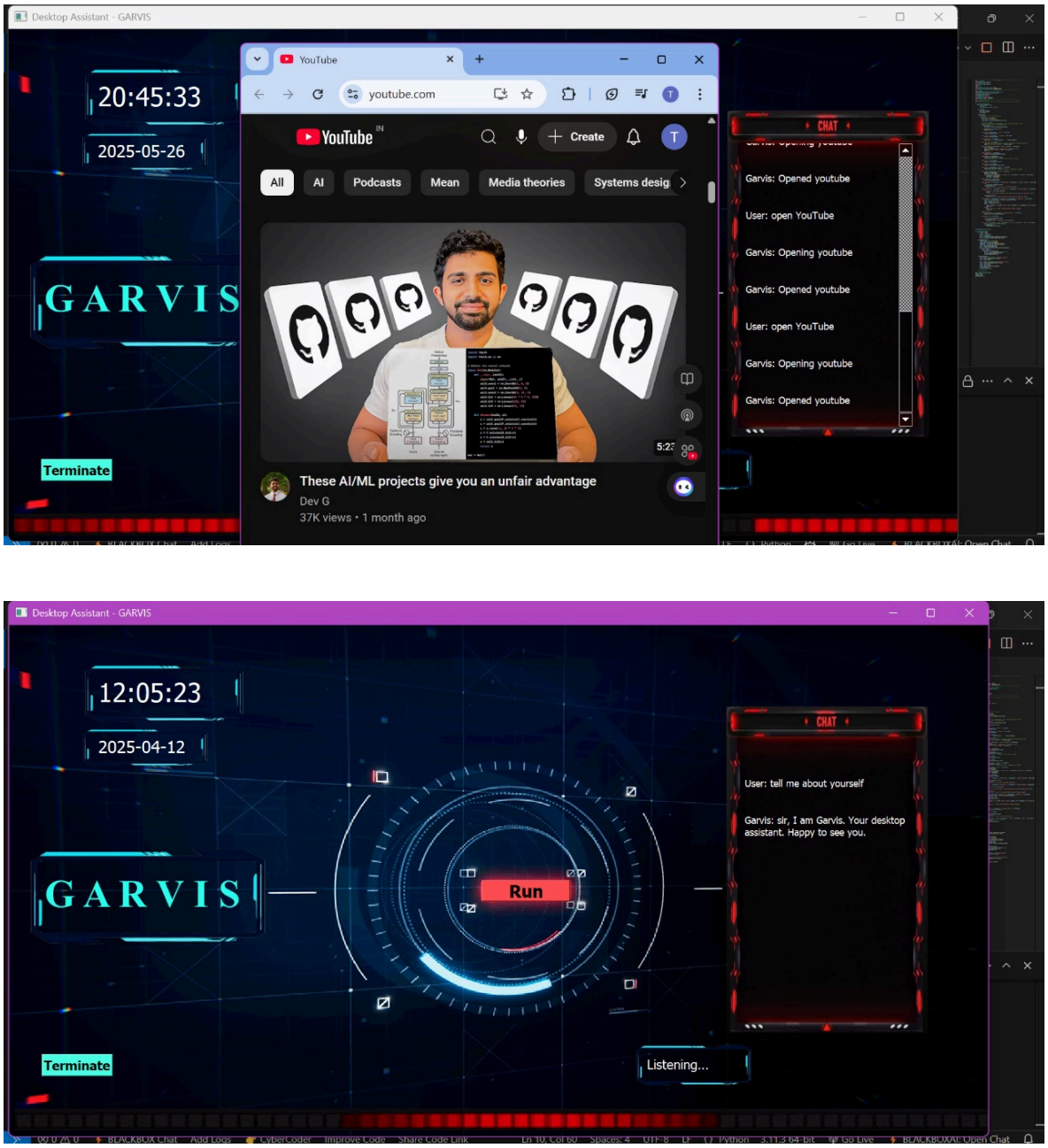


Figure 8.1: Results

The Desktop Assistant operates by bridging natural voice input with intelligent automation and response mechanisms. The seamless flow from speech recognition to task execution and result delivery enhances productivity and demonstrates the integration of AI, automation, and user interface in a single desktop solution.

CHAPTER 9

9.1 CONCLUSION

In conclusion, personal desktop voice assistants are becoming increasingly popular as more people seek convenience and efficiency in their daily tasks. With the development of advanced natural language processing and machine learning technologies, these assistants can understand and respond to human queries in a more intuitive and human-like manner. Personal desktop voice assistants have the potential to revolutionize the way we interact with our computers and devices, making it easier to navigate and access information. They can also enhance productivity and provide entertainment options, such as playing music or reading the news. While personal desktop voice assistants have many benefits, there are also some concerns around privacy and security. Users need to be aware of the data that is being collected and how it is being used to ensure their personal information is protected. Overall, personal desktop voice assistants have the potential to greatly enhance our daily lives and streamline our interactions with technology. As technology continues to advance, it will be interesting to see how these assistants evolve and improve in the years to come..

9.2 FUTURE WORKS

While the current implementation of Garvis demonstrates effective voice-based interaction and task automation, there are several areas where the system can be enhanced further. The following future enhancements are proposed to improve the functionality, intelligence, and user experience of the assistant:

1. Integration with IoT Devices

Expanding Garvis to control smart home or office IoT devices such as lights, fans, air conditioners, and smart appliances through voice commands using platforms like MQTT or Home Assistant can significantly increase its usability in real-world environments.

2. Enhanced Natural Language Understanding (NLU)

Incorporating advanced NLP models like BERT or integrating directly with GPT-based APIs can improve the system's ability to understand complex queries, follow up on context, and maintain conversations naturally.

3. Personalization and User Profiling

Adding user-specific customization features like voice-based authentication, remembering user preferences, scheduling reminders, and providing personalized suggestions can make the assistant more intelligent and user-centric.

4. Multilingual Support

Future versions of Garvis can support multiple languages, allowing users to interact with the assistant in their preferred regional language, enhancing inclusivity and accessibility.

5. Offline Capability

Currently, many functionalities rely on an active internet connection. Developing offline-capable modules for essential tasks like note-taking, application launching, and system control can ensure uninterrupted service even without internet access.

6. Emotion Detection and Sentiment Analysis

By integrating emotion detection using voice tone analysis or facial expressions (via webcam), Garvis can adapt its responses based on the user’s emotional state, making interactions more empathetic and human-like.

7. GUI-based Control Panel

A graphical interface can be introduced to offer users manual control alongside voice commands. This will make the assistant more intuitive, especially for visually impaired or differently-abled users.

CHAPTER 10

10.1 REFERENCES

Gong, L.: San Francisco, CA (US) United States US 2003.01671.67A1 (12) Patent Application Publication c (10) Pub. No.: US 2003/0167167 A1 Gong (43) Pub.

Date: 4 September 2003 for Intelligent Virtual Assistant Sarikaya, R.: The technology behind personal digital assistants. IEEE Signal Process. Mag.34, 67–81 (2017). https://doi.org/10.1109/msp.2016.2617341

Tsiao, J.C.-S., Tong, P.P., Chao, D.Y.: NaturalLanguage Voice-Activated Personal Assistant, United States Patent (10), Patent No.: US 7,216,080 B2 (45), 8 May 2007 4. Siri, K., Patankar, A.J.: Personal assistant with voice recognition intelligence. Int. J. Eng. Res. Technol. 10(1), 416–419 (2017). ISSN 0974-3154

Kawamura, T., Ohsuga, A.: Flower voice: virtual assistant for open data Elshafei, M.: Virtual personal assistant (VPA) for mobile users. Mitel Networks (2000– 2002)

Chung, H., Iorga, M., Voas, J., Lee, S.: Alexa, can I trust you? In: 2017 IEEE Computer Security (2017)

Cowan, B.R.: What can I help you with?: infrequent users' experiences of intelligent personal assistants. In: 2015 IEEE 10th International Conference on Industrial and Information Systems, ICIIS 2015, Sri Lanka (2015)

Weeratunga, A.M., Jayawardana, S.A.U., Hasindu, P.M.A.K, Prashan, W.P.M., Thelijjagoda, S.: Project Nethra - an intelligent assistant for the visually disabled to interact with internet services. In: 2015 IEEE 10th International Conference on Industrial and Information Systems (2015)

Lo pez, G., Quesada, L., Guerrero, L.A.: Alexa vs. Siri vs. Cortana vs. Google Assistant: a comparison of speech-based natural user interfaces. In: Nunes, I. (ed.) AHFE 2017. AISC, vol. 592, pp. 241–250. Springer, Cham (2018). https://doi.org/10.1007/978-3-319- 60366-7_23

Zhao, Y., Li, J., Zhang, S., Chen, L., Gong, Y.: Domain and speaker adaptation for Cortana speech recognition. In: ICASSP

Bellegarda, J.R.: Spoken language understanding for natural interaction: the Siri experience. In: Mariani, J., Rosset, S., Garnier-Rizet, M., Devillers, L. (eds.) Natural Interaction with Robots, Knowbots and Smartphones, pp. 3–14. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-8280-2_1

Google: Google Assistant. https://assisatnt.google.com

Purington, A., Taft, J.G., Shannon, S., Bazarova, N.N., Taylor, S.H.: Alexa is my new BFF: social roles, user satisfaction, and personification of the Amazon echo. ACM, 6–11 May 2017. ISBN 978-1-4503-4656- 6/17/05

Lopez, G., Quesada, L., Guerrero, L.A.: Alexa vs Siri vs Cortana vs Google Assistant: a comparison of speech-based natural user interfaces. Conference Paper, January 2018

Kepuska, V., Bohouta, G.: Next generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa, and Google Home). In: IEEE Conference (2018)